

**Version 1.1**

**Date: 2018/12/09**

Abstract

**The objective of this document is to describe the Catalogue Maintenance application that supports structures and applications driving the sales decision support business requirements**

Catalogue Maintenance

*Define All Structures and Application supporting lookup demands*

**Table of Contents**

Document approval and distribution list 2

1. Introduction 3

2. Audience 3

3. Objectives 4

4. Business Flow 4

5. Detail description of functionality 4

6. Dependencies 5

7. Design philosophy 5

8. Database design approach 6

9. The advanced searching approaches 8

10. Catalogue lookup to sales-order 9

11. Database entities and relationships 11

12. Programs 13

12.1 MS Windows Executables 13

12.2 SQL Stored Procedures 13

13. Acceptance 14

# Document approval and distribution list

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Name / Title** | **Signature** | **Date** |
| **Document Type / purpose** | | | |
| Prepared by |  |  |  |
| Reviewed by |  |  |  |
| Approved by |  |  |  |

# Introduction

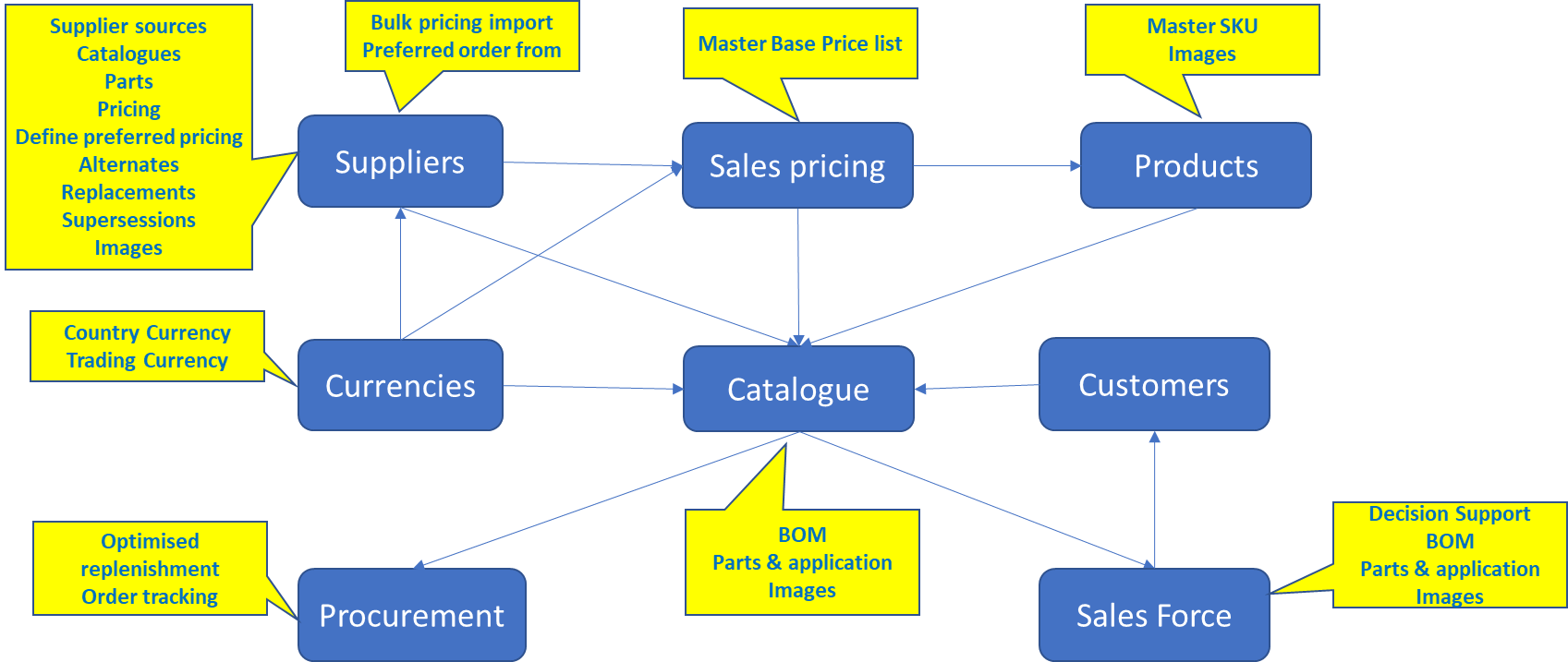
The so-called **Catalogue Lookup**represents an end user tool that is able to promote sales of parts to the industry through systems-based intelligence that is created using data captured in various sub systems in ePart.

Systems functionality exists to consolidate the data in contained in the various subsystem in an optimised manner in support of business requirements.

Additionally, there is functionality to create assemblies and sub-assemblies that represent the construct of parts and their relevant applications.

The following diagram provides a high-level end to end view of the ePart Catalogue system with some indicative functionality guideline annotations relative to sub-system functionality.

**Diagram 1.1: *A high level view of the ePart catalogue sub-system integration***



There is no ***redundant data in ePart due to the end to end systems and application integration***, consequently when reviewing the ePart ***catalogue*** end to end, it may lead to a misconception that it is complex. However, by brining all the sub-system components together in a unified manner it exposes the inter sub-system dependencies properly and negates the need to re-capture the same data repeatedly.

About the ***catalogue*** maintenance section, there are a number of functions that ease the repetitive work of catalogue engineers; some of these are:

* Creating tree structures of assemblies and sub-assemblies in line with the manufacturing industry and resembles the motor manufacturing industry method
* Creating assembly templates that can be re-used within the assembly tree
* Inheritance of template assemblies onto parent assemblies with or without future change inheritance.
* Example of this is the Ford 1000cc engine used in more than one vehicle. The engine assembly is created as a template and then linked onto the various vehicles where it is used. After linking, the linked version can be changed for subtle variances specific to the linked to vehicle. Although the example is for vehicle and engine, the same principal can be used at any level within the tree structure.
* Assemblies may be SKU items that can be sold i.e. turbo charger. However, a turbocharger may have sub-assemblies that could be selling SKU’s as well i.e. shaft bearings. Thus, the tree allows for the opportunity to define sellable SKU items as well as the constituent components that make up the SKU assembly.
* It must be noted that one or more images can be added at SKU level and at assembly level in the tree. The assembly level image would be applicable where an image would assist the selling process for the question ‘Does the bearing fit this gearbox?’ and we do not stock gearboxes.

# Audience

Sales

Cataloguing

Management

# Objectives

To optimise the definition process of creating structures and relationships that depict the ***real-world*** of stocking items and their related applications. In support of this, the database structures are defined using an ***inverted tree structure to manage the so-called bill of material definition.***

Follow the real-world parts identification allowing these to resolve back to the ePart base number (SKU)

Provide parts and assembly images to assist in the identification process, both for purchasing and sales teams.

Include as much detail as is possible so that operationally the business is optimised through the availability of pertinent information at the time. Illustratively, tariff heading in support of imported goods as well as for the export to neighbouring countries.

Have relevant dashboards through which the underlying data can be observed, and management decision made to improve business processes i.e:

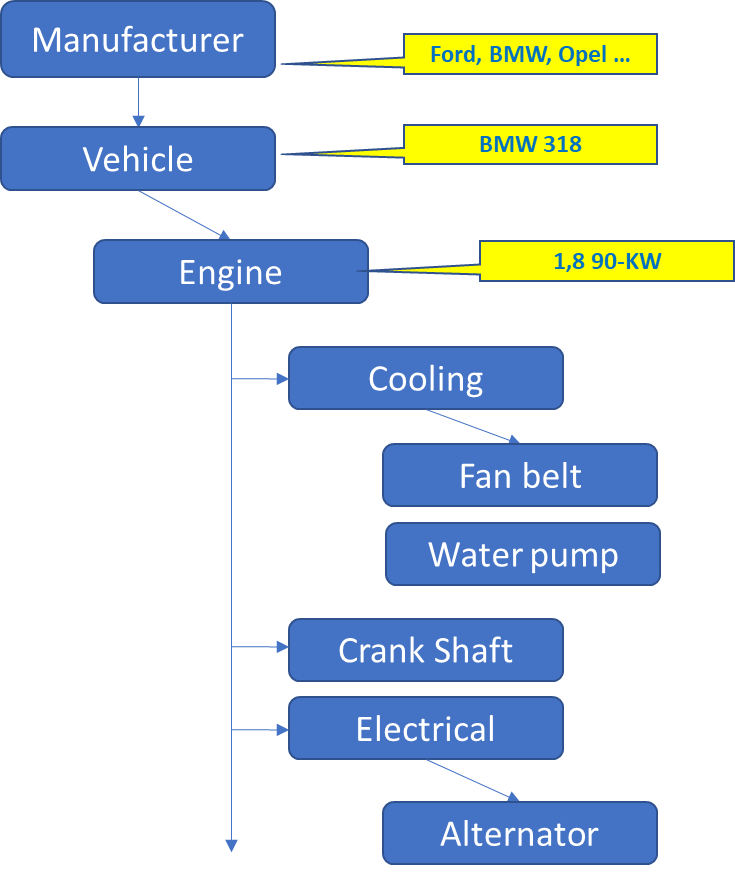
* Selecting best supplier option for replenishment
* Define which supplier pricing to be used when calculating standard selling price
* Define which supplier pricing to be used when calculating wholesale price lists
* View sales trends to optimise pricing overall
* ….

# Database Design Objectives

Analysis of the data available, data gathering from various sources, relationships of parts to applications, it became evident that a ***formalised hierarchical structure*** (inverted tree) would be important in the data persistence implementation to support operational functionality.

Contained in the legacy (COBOL based) catalogue that was to be replaced, was details of some part to application details. Although simplistic in representation, there was enough textual detail in the parts description to infer an inverted tree representation which was thought to be a suitable fit for the data available.

Some of the historical experience in the developer team was to develop solutions with open ended parent child relationships which does solve the visual view of the parts to application relationships in the following manner:

**Diagram 4.1: *Provides a high-level view of the hierarchical data representation needed for parts to application relationships***

The following simplified diagram illustrates the database implementation that supports the inverted tree BOM implementation with linking to parts (SKU) as part of the leaf nodes:

**Diagram 4.2: *Inverted tree with leaf node parts linking***

This ***self-referencing*** relationship provides for unrestricted levels of relationships to be assembled in an ***assembly*** to ***Sub-Assembly*** without limits. The ***Part(s)*** relationship provides the ***item level component stocking attributes*** as depicted in the following diagram:

**Diagram 4.3 provides a view of how the relationship is used to depict an easily viewed construct**

The assembly / sub-assembly explosion indicates how elements with related attributes can be viewed with the highlighted entry indicating the actual stock item.

By design the solution provides for the linking of assemblies to stocking items even when the stock item is made up of sub-assemblies i.e. a turbo charger can be sold as a complete unit yet some parts making up the turbo charger can be sold separately.

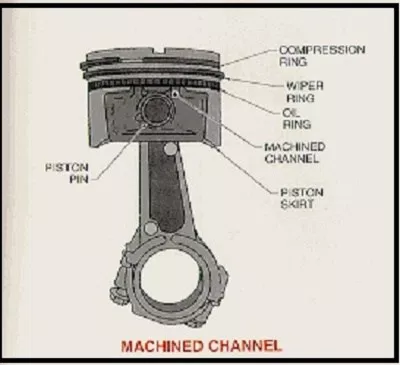
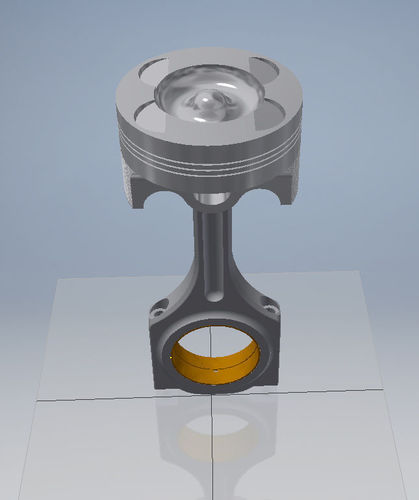
***Note that there are more control attributes such as assembly type, code and others. Refer to the full entity relationship chart for more details***

# Data normalisation

When the database design was undertaken, the understanding was gained that the traditional data normalisation (5th normalisation) against each product type could not succeed against the variety of SKU types in existence. This in support when searching for a suitable bearing size vs a piston of a type.

By way of an example, a piston as a type has properties that are generally repeatable between pistons of different piston applications i.e. attributes types of the following:

* Diameter
* Skirt length
* Compression rings
* Wiper rings
* Oil rings
* Gudgeon pin size
* ….

On the other hand, ball bearings also have repeatable attributes between ball bearing types with attributes as follows that do not match those of a piston:

Speed Ratings

Reference speed: 75000 r/min

Limiting speed: 48000 r/min

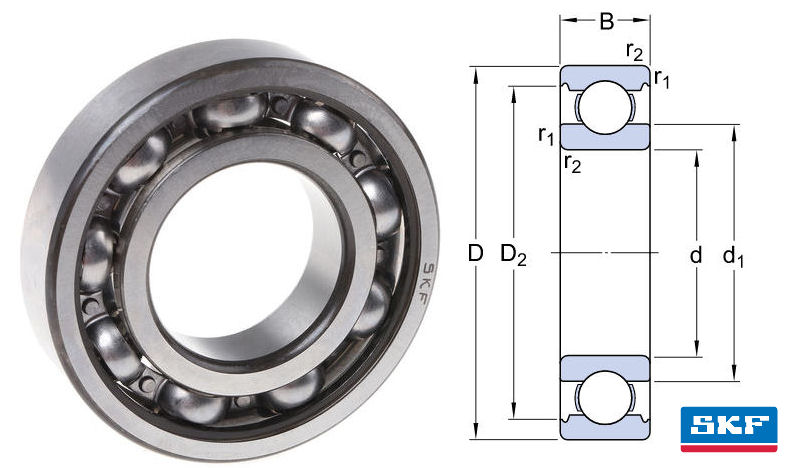
Dimensions mm

d1: 12.1

D1: 17.6

D2: 19.2

r1,2 min: 0.3



From a technical perspective the vast number of SKU (base numbers / parts) make a standard data normalisation approach an almost impossible task.

The design team of the day opted to allow the SKU properties to be user definable where a SKU is assigned a type and one or more properties against the type defined. Consequently, the data capturing program is SKU type aware and will limit the user input to the defined properties. This removed the need to perform data normalisation on a formalised manner but retained the ability to perform lookup searching against the defined attributes as well

# Multiple supplier for same part

As part of the catalogue maintenance sub-system is to work through parts catalogues received from various manufacturers / suppliers. Details pertaining to the items listed in the catalogue are captured or updated without necessarily linking said part number to a system SKU (base number). It is assumed that at least that if there is a reference it will trigger a need to do further research to identify exactly the part and its application.

If at any time the un-linked part is identified against a SKU (base number) or another supplier part reference which is linked, the un-linked item is automatically cascade linked using the qualified detail exposed.

This feature allows purchasing staff to identify alternate suppliers from whom to order from.

It is a reality that OEM allocate part identifications per their Design & Manufacturing process that is not used by replacement part manufacturers – they use their own part identification system. However, the alternate part manufacturers do provide OEM referencing.

In support of this, ePart catalogue maintenance allows the creation of alternate identification part numbers from various suppliers / manufacturers. These linked back to the ePart Base number (SKU). Should a query be launched using the catalogued industry part reference, the ePart Base number can be resolved and hopefully the business concluded with success.

In the case of non-local suppliers (import), enough detail can be captured to allow for the export documentation to be completed i.e. tariff headings amongst others. Additionally, the product industry application detail (agricultural, commercial etc) needs to be maintained for government purposes so that import duties can be accounted for.

As part of the captured data, supplier parts profile is catered for i.e. dims, weight etc. Also, and of importance is the opportunity to capture the container bar code reference assigned to the supplier part. This bar code is linked to the SKU (Base Number) and can result in eliminating the print requirement for the ePart barcode.

Supplier bar codes can be captured as well allowing for speedier receiving

(Note: a Scoping document has been produced and needs management approval to execute on)

# Alternates, supersessions and replacement parts

As the assigned engineering skilled staff review their catalogue data, provision is made to mark parts in the following manner:

* Alternates is where a part is not the same but is otherwise a good alternate to use i.e. fan belts may have differing lengths but can do the same work.
* Supersessions is where a part is superseded for some reason by a newer part, it may very well be that the both the superseded and supersession are still serviceable. However, in the ePart system there is a specific flag setting that indicates that the superseded part may not be sold and will stop a sales transaction.

# Dependencies

|  |  |  |
| --- | --- | --- |
| # | Description | Action / By whom |
| 1 | Accounts receivable |  |
| 2 | Pricing |  |
| 3 | Catalogue maintenance |  |
| 4 | Discount structures |  |
| 5 | Warehouse activities |  |
| 6 | Sales order |  |

# Design philosophy

The design philosophy closely adheres to the manner in which the ePart was incarnated by splitting the entire application into 3 basic components:

* 1. Presentation – this is done in Builder c++ with limited if any engagement of business logic
  2. Business logic – this is done using MSSQL stored procedures
  3. Data persistence – the fact that business logic is contained in Stored Procedures makes the persistence integral part of the business logic process.

The underlying benefit to this approach is simplicity and maintainability.

The obvious notion to this is that the presentation logic can theoretically be swapped out retaining the business and persistence logic intact.

# Database design philosophy

In the after-market automotive industry the concepts of assemblies, sub-assemblies and finite parts are strongly represented.

Consequently, the ePART catalogue contains all parts in relation to a tree of assemblies, called the bill of materials, or BoM.

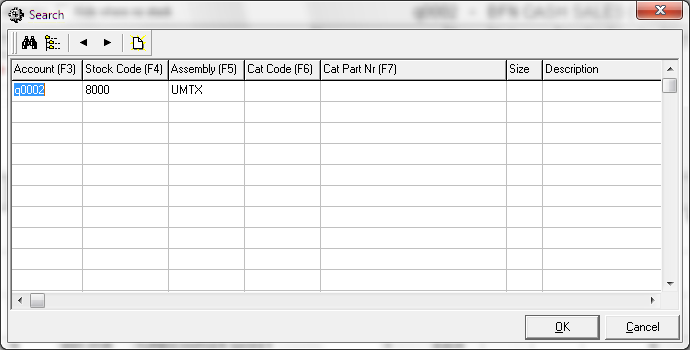
**Diagram 8.1 provides a high-level view of the *core* catalogue Entity Relationship as deployed**

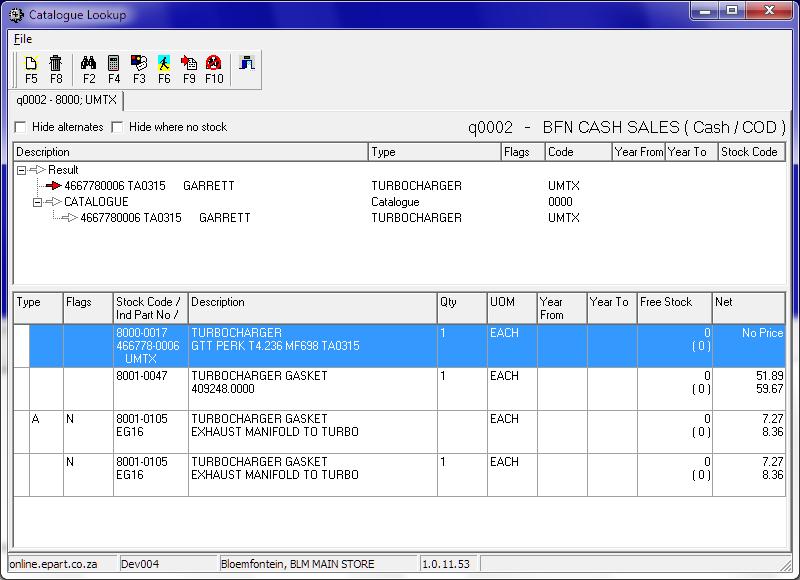


# The advanced searching approaches

In addition to the parts explosion (inverted tree) format of searching, there is the advanced searching capability and is depicted in the following diagram:

**Diagram 9.1: A high-level view of advanced search capabilities:**



**Diagram 9.2: The search results:**

Some of the search criteria can expose the concept of ***KITS*** being for servicing or engine overhaul and several others as well. This allows for opportunistic selling by the sales staff enhancing the customer experience.

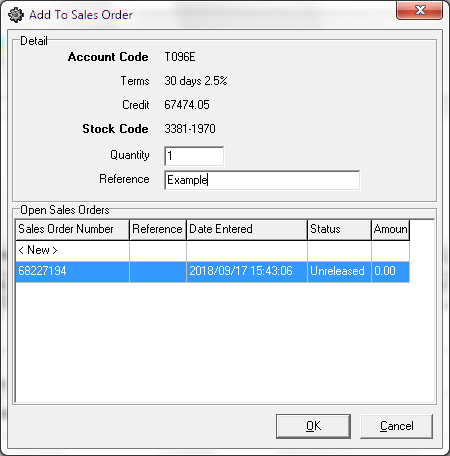
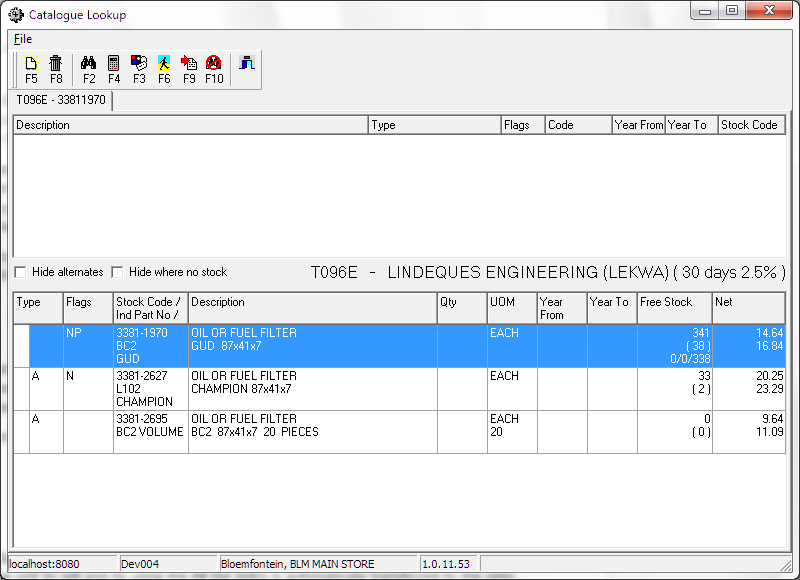
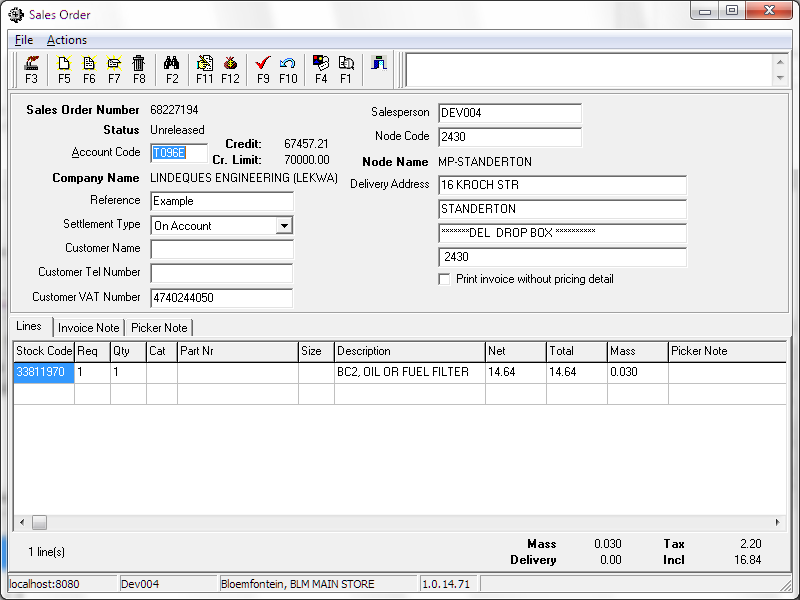
The catalogue lookup allows for the searching using industry part numbers or supplier part numbers for the same stocking unit. This is for instance a piston ring can have multiple industry part numbers depending on which manufacturer distributer is called off by the customer.

To enhance customer experience, the system allows for supersessions, alternates and discontinuations.

# Catalogue lookup to sales-order

The catalogue lookup application operates independently from the sales order program. However, there is a path of communication from the catalogue lookup to the sales order as depicted in the following diagram

**Diagram 10.2: A view of how the lookup interacts with the sales order**



The effective optimisation is that from the search and filter results set, a choice is made of the stocking unit to sell and by using the ***F9*** the entry is automatically transferred to the sales order.

# Database entities and relationships

The full catalogue related database diagram provides a view of the various participating relationships. Notably is the simplicity of the database tables participating in the core structure.

Many of the additional tables are there for control and optimisation purposes.

A specific reference is made to the following tables:

* “Bridging table” – this is often used in data mining structures where the volume of data requires a specialised table of this kind
* BOM level – this table is a control table to prevent illogical parent child linking i.e. linking an engine to a water-pump rather than linking a water-pump to an engine
* ….
* ….
* ….
* …

The catalogue support system provides for a highly efficient source of information, provided the underlying data is accurately maintained and kept up to date; failing which will result in poor customer experience.

# Programs

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| catLookup2.exe | Used to look up which parts the customer wants to purchase, and add them to the sales order. |

# SQL Stored Procedures

|  |  |
| --- | --- |
| **Name** | **Description** |
| catCatQueryResultHomeFindByQueryNo2 | Searches for items matching the criteria, to populate the item grid. |
| catCatQueryBomNodeHomeFindByQueryNo | Finds the section of the BoM above and below the search results. |
| salAddToSalesOrder | Automatically adds an item to an existing or new sales order. |
| catLogLostSaleManually | Records a formal lost sale, in the form of a failed-search-type informal lost sales marked as “manual”. |

# Acceptance

I hereby confirm that I have been fully informed of the document’s content and received training to understand how the detailed instructions are to be applied:

Name …………………………………………………………………………….

Job Title ………………………………………………………………………….

Signed ……………………………………………………………………………

Date ………………………………………………………………………………